Adaptive Domain Decomposition for Neural Surrogate Model Compression

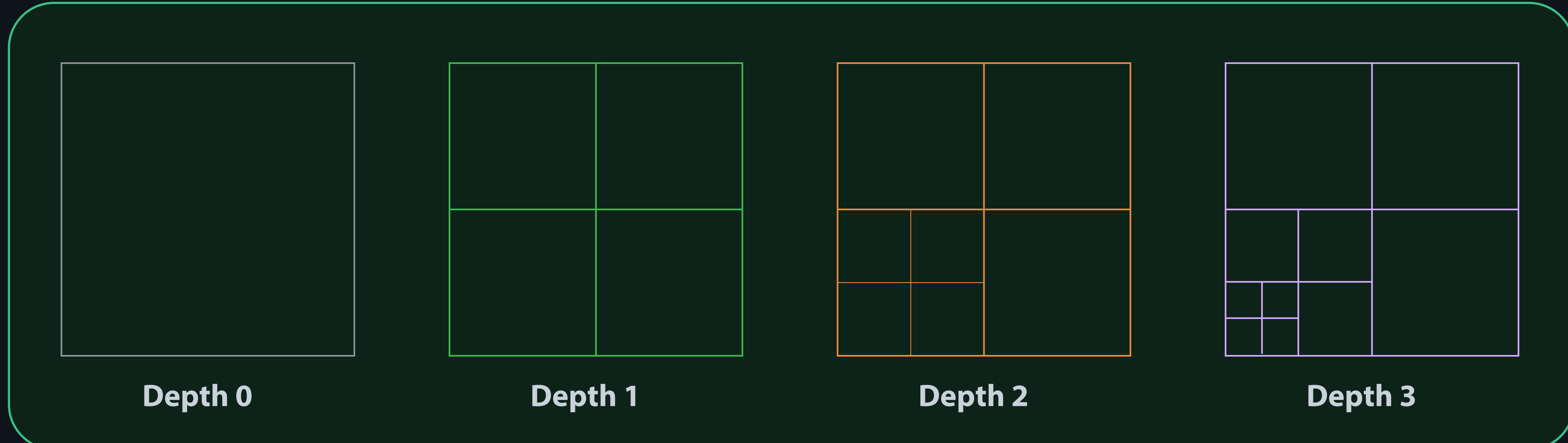
VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER
www.it4i.cz

Marcel Smolík
Tomáš Brzobohatý

Problem and Motivation

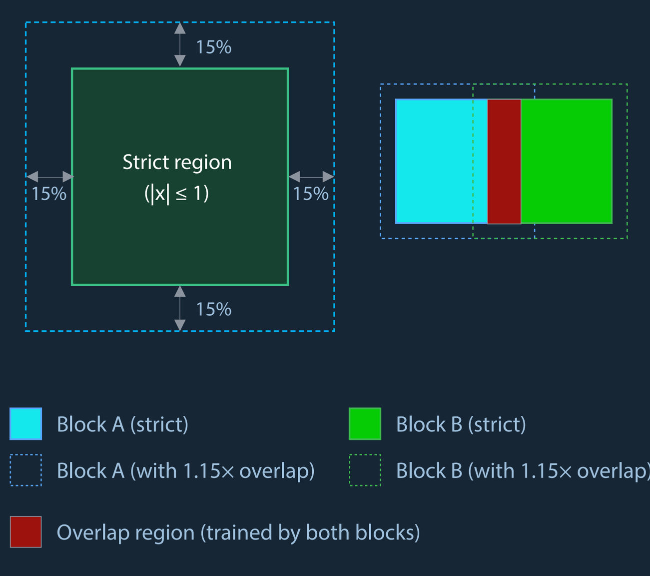
Modern CFD simulations generate terabytes of high-dimensional spatiotemporal data, creating major challenges for storage, transfer, and real-time analysis. Neural surrogate models offer a promising pathway for extreme data compression; however, conventional monolithic neural architectures suffer from spectral bias, capturing only low-frequency flow behavior while failing to preserve localized turbulent and chaotic structures on large computational meshes. To address this limitation, we propose a scalable neural compression framework based on overlapping domain decomposition. The computational domain is recursively partitioned into smaller subdomains, where independent neural networks locally learn the flow dynamics with adaptive model capacity. Overlapping regions are reconstructed using Gaussian-weighted blending, ensuring smooth continuity across subdomain boundaries. This distributed architecture enables parallel training and inference, significantly improves scalability on large CFD problems, and preserves high-frequency transient physics while reducing massive simulation datasets from gigabytes to megabytes.

Adaptive Octree Domain Decomposition



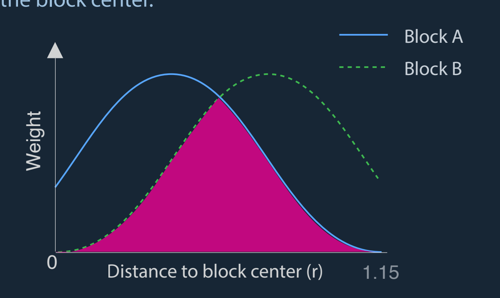
OVERLAPPING BLOCKS (1.15×)

Each block extends its strict domain by 15% in every direction.
This overlap ensures smooth transitions between neighboring blocks.



GAUSSIAN BLENDING

In overlapping regions, predictions from multiple blocks are blended using Gaussian weights based on distance to the block center.



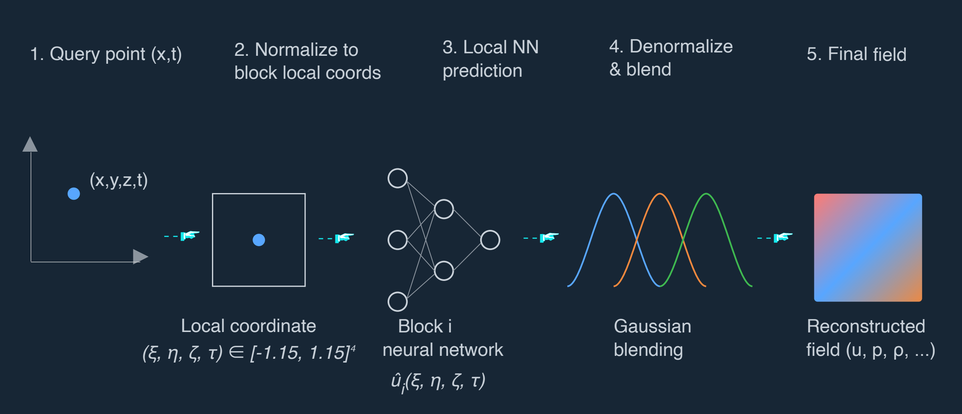
Gaussian weight for block i:

$$w_i(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$$

Blended prediction:

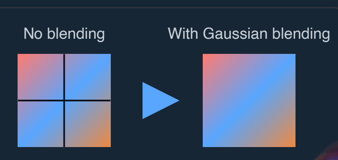
$$\hat{u}(\mathbf{x}, t) = \frac{\sum_i w_i(\mathbf{x})\hat{u}_i(\mathbf{x}, t)}{\sum_i w_i(\mathbf{x})}$$

FULL 4D RECONSTRUCTION PIPELINE



Why blending matters?

- Ensures continuity across block interfaces
- Reduces boundary artifacts
- Provides stable and accurate global reconstruction



Stopping criteria for a block:

- Number of points < min_points
- or depth ≥ max_depth

Each leaf block stores:

- Center, scale, depth
- Indices of points in an OVERLAP region (1.15×)

- CFD data were computed using the Lattice Boltzmann Method
- ProLB - www.prolb-cfd.com
- The simulation represents compressible flow around a 3D sphere.
- The flow regime corresponds to Re = 10,000.
- The compression dataset contains 50 time steps.
- The compressed fields include pressure, velocity, and density.

Table 1: Reconstruction Fidelity Metrics by Architecture

Architecture	Field	Cent. Err (%)	SNR (dB)	PSNR (dB)	Pearson R
SIREN (3 Layers)	Velocity	2.495	36.0	50.6	0.9999
	Pressure	1.300	37.7	59.6	0.9999
	Density	1.296	37.7	59.7	0.9999
SIREN (4 Layers)	Velocity	1.740	39.1	53.8	0.9999
	Pressure	0.927	40.7	62.6	1.0000
	Density	0.922	40.7	62.6	1.0000
Fourier (3 Layers)	Velocity	4.635	30.6	45.3	0.9996
	Pressure	3.020	30.4	52.3	0.9995
	Density	3.021	30.4	52.3	0.9995
Fourier (4 Layers)	Velocity	3.932	32.0	46.7	0.9997
	Pressure	2.363	32.5	54.5	0.9997
	Density	2.361	32.5	54.5	0.9997

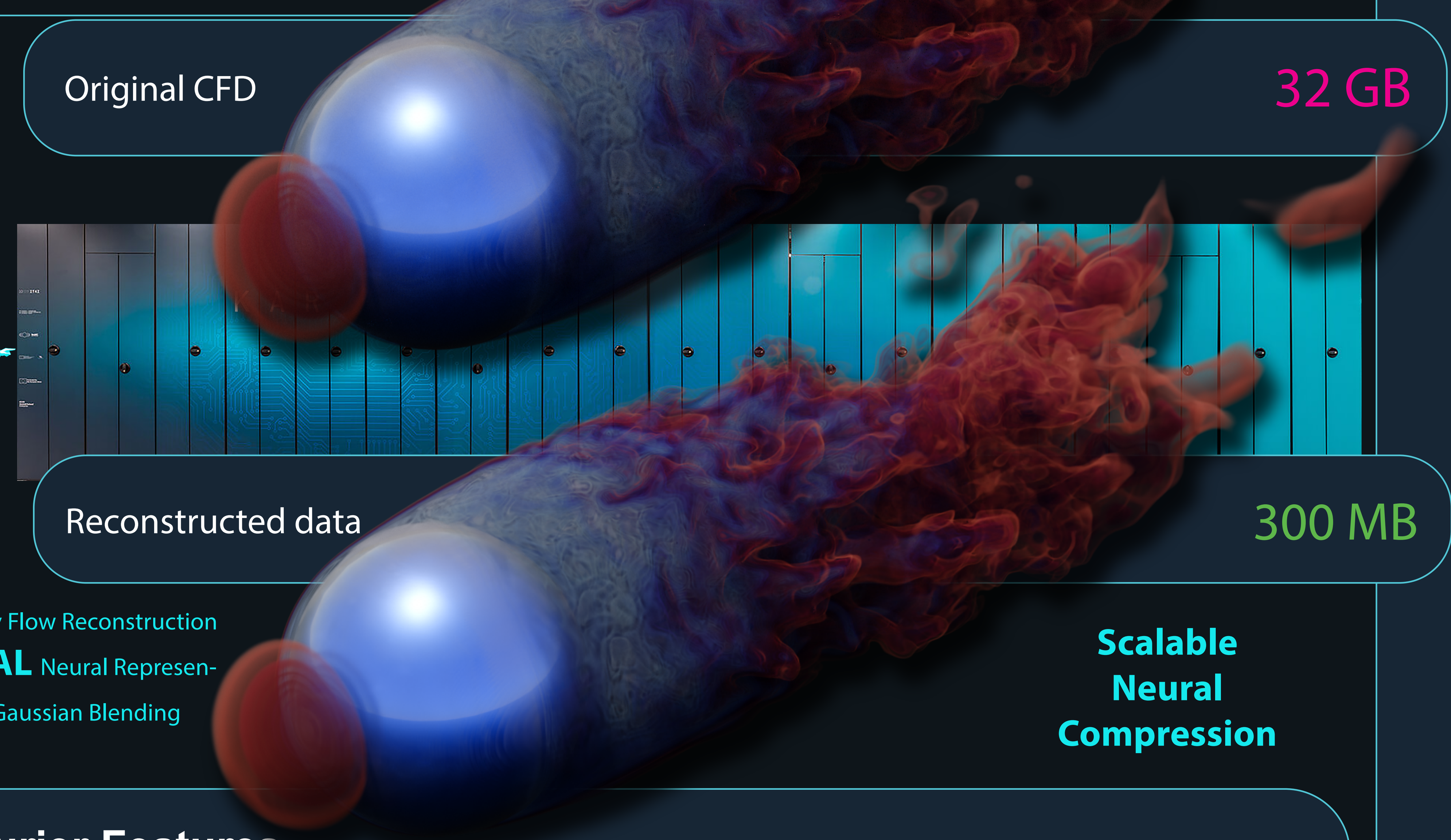


Table 2: Computational Performance and Compression Efficiency

Architecture	Model Size (MB)	Comp. Ratio	Throughput (10^6 pts/sec)	Train Time (64 GPUs)
SIREN (3 Layers)	6.82	89.13×	4.77	18:30
SIREN (4 Layers)	9.44	64.41×	4.04	23:40
Fourier (3 Layers)	12.54	48.51×	4.59	19:20
Fourier (4 Layers)	14.38	42.30×	3.58	24:20

High-Fidelity Flow Reconstruction with **LOCAL** Neural Representations and Gaussian Blending

Scalable Neural Compression

Fourier Features

Model: DynamicFourierMLP (spatial_dim = 4, out_fields = 5, freqs = 32, base_freq = 1.00)

Input (x_in)
Shape: (N, 4)
N = batch size

(spatial)
x_1
x_2
x_3
x_4

FourierEncoding
(input_dim = 4, num_frequencies = 32, base_freq = 1.00)

Input
x
Shape: (N, 4)

For each frequency fk:
k = 0 ... 31

$\sin(2^k \pi x)$
Shape: (N, 4)

$\cos(2^k \pi x)$
Shape: (N, 4)

Concatenate
all frequencies
with original x
Shape: (N, 260)

Output dimension = input_dim + 2 * input_dim * freqs = 4 + 2 * 4 * 32 = 260

MLP (Sequential)

Linear(260 → 32)
GELU

Linear(32 → 32)
GELU

Linear(32 → 32)
GELU

Linear(32 → 32)
GELU

Linear(32 → 5)

Output
(predicted fields)
Shape: (N, 5)
f_1
f_2
f_3
f_4
f_5
(out_fields = 5)

Fourier Encoding Details

- input_dim = 4
- num_frequencies = 32
- frequencies: $\pi \cdot 2^k$, k = 0 ... 31
- Output dim = 4 + 2 * 4 * 32 = 260

MLP Architecture

- Input dimension: 260
- Hidden layers: [32, 32, 32, 32] (width = 32)
- Activation: GELU
- Output dimension: 5
- Total linear layers: 5 (4 hidden + 1 output)

Dynamic Scaling

- volume_depth = depth // 3
- width = 32 (constant in current config)
- base_freq = 1.00
- freqs = 32
- spatial_dim = 4, out_fields = 5

Training Components

- Loss: SmoothL1Loss(beta = 0.1)
- Optimizer: Adam (fused), lr = 1e−3
- Scheduler: StepLR, step_size = 1000
- gamma = 0.5
- Epochs: 1000

Key Dimensions

Input: (N, 4)
After Encoding: (N, 260)
Hidden Layers: (N, 32)
Output: (N, 5)
N = batch size

Note: Fourier Feature mapping lifts the 4D input into a high-frequency space, followed by an MLP to regress 5 output fields.

SIREN Architecture

Sinusoidal Representation Networks for 4D Coordinate Regression

Input
$x \in R^{N\times 4}$
(Batch of N points)
4D coordinates
(t, x, y, z)

SIREN Network (w_0 = 30.0)

Layer 0
(SineLayer - First)
Linear(4 → 256)
$\sin(w_0 \cdots)$
Shape: (N, 256)

Layer 1
(SineLayer)
Linear(256 → 256)
$\sin(w_0 \cdots)$
Shape: (N, 256)

Layer 2
(SineLayer)
Linear(256 → 256)
$\sin(w_0 \cdots)$
Shape: (N, 256)

Layer 3
(SineLayer)
Linear(256 → 256)
$\sin(w_0 \cdots)$
Shape: (N, 256)

Layer 4 (Output)
Linear(256 → 1)
No activation
Shape: (N, 1)

Output
$\hat{y} \in R^{N\times 5}$
Shape: (N, 5)

SineLayer Details

x → Linear (W x + b) → Sine Activation ($\sin(w_0 \cdots)$) → y

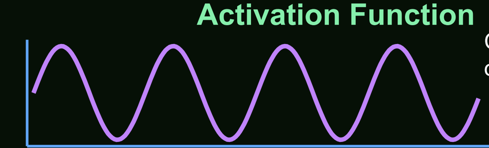
Weight Initialization

First Layer: W ~ U(−1/in, 1/in)

Hidden: W ~ U(−√(6/in)/w_0, √(6/in)/w_0)

Final linear layer uses the same scaled initialization, without activation.

Activation Function



Controls frequency of the sine activation
w_0 = 30.0

Tensor Shape Flow

- Input → (N, 4)
- Layer 0 → (N, 256)
- Layer 1 → (N, 256)
- Layer 2 → (N, 256)
- Layer 3 → (N, 256)
- Layer 4 → (N, 1)

Architecture Summary

- Input dimension: 4 (t, x, y, z)
- Hidden layers: 4
- Hidden width: 256
- Activation: $\sin(w_0 \cdots)$, w_0 = 30
- Parameter count ≈ 4.2M

Key Characteristics

- Uses sinusoidal activations instead of ReLU/tanh
- Captures high-frequency signals naturally
- Suitable for implicit neural representations
- Requires input coordinates scaled to [−1, 1]
- Sensitive to the frequency parameter w_0

Training Notes (from code)

- Loss: Mean Squared Error (MSE)
- Optimizer: Adam, lr = 1e−4
- No learning rate scheduler
- Inputs scaled to [−1, 1] range

Intuition: SIREN represents signals as learned sinusoidal components, enabling fine high-frequency CFD detail reconstruction.